Extending CPN Tools with Ontologies to Support the Management of Context-Adaptive Business Processes

Estefanía Serral, Johannes De Smedt, and Jan Vanthienen

KU Leuven Faculty of Economics and Business
Department of Decision Sciences and Information Management
Naamsestraat 69
B-3000 Leuven, Belgium
firstname.lastname@kuleuven.be

Summary. Colored Petri Nets (CPN) are a widely used graphical modeling language to manage business processes. Business processes often appear in dynamic environments; therefore, context adaptation has recently emerged as a new challenge to explicitly address fitness between business process modeling and its execution environment. Although CPN can introduce data by defining internal data records, this is not enough to capture the complexity and dynamics of the execution context data. This paper extends CPN tools to support the management of context-adaptive business processes. To achieve this challenge, CPN tools are integrated with ontology-based context models that properly represent and manage the business process context. This allows context to be appropriately modeled at design time, and queried and updated at runtime. The combination of ontologies with CPN tools presents a way to bridge business processes management with context data management while treating data and behavior as separate concerns. In this way, system design, reuse, and maintenance are also improved.

Key words: Colored Petri Nets, Context Adaptivity, Ontologies, CPN tools.

1 Introduction

Colored Petri Nets (CPN) [1] are a mathematical modeling language that has a graphical notation and very powerful analysis techniques. For these reasons, the importance of this language is undeniable in Business Process Management (BPM) [2].

Since users and organizations as well as their software systems operate more and more in dynamic environments, context adaptation has recently emerged as a new perspective in BPM to explicitly address fitness between business process modeling and its execution context. Thus, it is made possible that the behavior in the information system automatically adapts in order to effectively operate in environments where context changes frequently. In BPM, context is usually

defined as the set of properties that can influence and change business process execution, properties that are normally highly dynamic, such as: equipment state and location, time, season, temperature, stakeholders' location and preferences, etc. [3].

CPN support the use of data variables as internal data structures, and the assignation of values as tokens embedded inside the net. For instance, Figure 1 shows a workflow example of a product sale in a shop using CPN. Sellers (SEL), Customers (CUST), and Products (PROD) are color sets represented by an id. The tokens of these colour sets are indicated as input in the corresponding net transitions. In the process, a seller approaches a customer and explains him/her a product. The customer can either buy it or leave the shop. The seller will be redirected to the seller pool after a successful sale or when the customer leaves. If a products gets sold, its registry is updated; otherwise, it returns to the product pool.

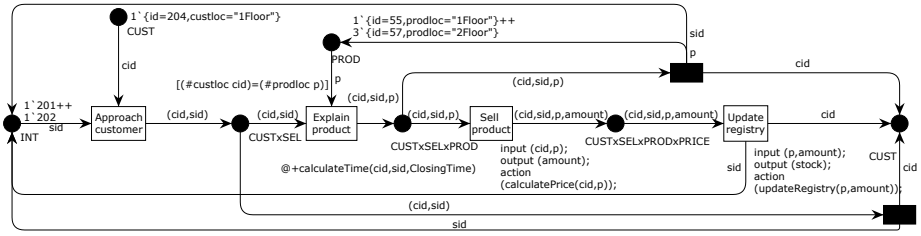

Fig. 1: Example workflow of a shop.

Although CPN provide great expressivity for representing the process behavior, it is not enough to represent and manage the execution context dynamics and the context adaptive behavior. This would support the use of Petri nets in dynamic environments and will allow the definition of business processes that adapt to their execution environment. For instance, by using context, it would be possible to use information regarding the current location of the customers, sellers and products to make a better match and increase the chances of selling products. Also, the products could be promoted depending on context such as the timing, the market, the season, etc.; i.e., heating devices could be promoted in unexpected colder periods.

In addition, in order to facilitate system design, reuse, and maintenance, context data must be specified independently from the Petri net behavior ensuring an effective separation of concerns. In this paper, we extend CPN tools [4] and integrate them with ontology-based context models to properly support context adaptation in CPN. By using ontologies, it becomes possible to represent context data with a high degree of expressiveness, to update it at runtime, and to incorporate it in multiple workflows throughout the system. While previously all tokens and variables needed to be specified using basic data structures

and needed to be process model specific, this approach lifts the workflow data management to a higher level.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 explains the extended CPN to describe context-adaptive business processes. Section 4 describes the developed tool to support the extension. Finally, Section 5 discusses the presented contributions and further work, and Section 6 presents the conclusions.

2 Related Work

In [3], [5], [6], the different kinds of business process context properties are analyzed. These works acknowledge the need of analyzing business process context to make BPM more effective. However, they only focus on identifying which context properties are relevant, and they do not provide support for context adaptation. Also, works such as [7], [8] emphasize the need for process flexibility through the support of adaptations during process execution; however, none of them focus on providing context adaptation in Petri nets.

Several approaches have been proposed to extend Petri nets in order to support context adaptation in business processes. Lu et al. [9] present a scenario-based method to design context-aware service models in which Petri nets are used as the formalism for building a logic model of context-aware services. Petri nets are extended with context functions that can be linked to states or transitions. This work focuses on system design, not paying attention to how context should be represented and managed to be used in order to adapt the Petri nets execution.

Feilong Tang et al. [10] present a context model and a context-aware workflow management algorithm for ubiquitous campus navigation. They extend Petri nets with the context concept; however, the proposed context model and execution algorithm are tied to the campus navigation application and the integration of the context concept in Petri nets is rather abstract.

Ardissono et al. [11, 12] present a framework for the context-aware management of applications based on the composition of Web Services in complex workflows. The context-aware workflow execution is based on the introduction of abstract activities. Each abstract activity has an associated set of context-dependent implementations representing the alternative courses of action that the workflow engine should select depending on the context. Thus, instead of representing context adaptation in Petri nets, each alternative is represented using a different Petri net. In addition, no detail is given about how context is specified and managed.

Although these works advance the state of the art towards supporting context-adaptive business processes with Petri nets, none of them provides either a proper support for context management or any tool support for managing context-adaptive business processes using Petri nets.

3 Modeling Context-adaptive Business Processes

We integrate two models to represent context-adaptive business processes: a) an ontology-based context model, where the business context is semantically described; and b) an extended CPN where the business process can refer to the context described in the context model.

3.1 Context Modeling

In order to describe the business context, we use ontologies since several studies [13, 14, 15] state that the use of ontologies to model context is one of the best approaches for this purpose. Ontologies guarantee a high degree of expressiveness, formality and semantic richness, and exhibit prominent advantages for reasoning and reusing context as well as facilitating the interoperability of different systems.

An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. An ontology mainly contains the following elements:

- **Classes:** all kinds of entities or concepts. A class usually refers to a collection or a category of objects sharing some common character and well accepted under common sense; e.g., the classes product and location.
- **Data properties:** properties that identify a class itself from other classes and has a basic type, such as int, string, time, etc.; e.g., name and age.
- **Object properties:** properties that identify a relation between two ontology classes, i.e., identifies how an object is connected to other object in an ontology, e.g., a product_location, which relates the product class and the location class.
- **Constraints:** rules that must be satisfied for the elements for which are defined; e.g., the cardinality of a certain property must be 1; the class A is subclass of the class B; the object property is_in (which relates one location object to other location object) is transitive (i.e., if a location X is in the Y location and the Y location is in the Z location, then, X is inside Z too); etc.
- **Individuals:** instances or objects of the defined classes; e.g., the individual Dell_latitud_e6410_16 of the class product, with name Dell_laptop, and which product_location is office236, which is an individual of the location class.

In order to specify the context of the application, first the context classes, properties, and constraints for the domain should be identified (this should be reusable for each application within the domain). Afterwards, the specific context application should be specified as individuals of the defined classes.

To represent the ontology-based context model, we use the Web Ontology Language (OWL)[1]. OWL is a machine-interpretable semantic markup language for publishing and sharing ontologies and is an open World Wide Web Consortium (W3C) standard. OWL is designed to provide a vocabulary along with a

[1] http://www.w3.org/TR/owl-ref/

formal semantics in order to facilitate the management and processing of knowledge at runtime. Thus, it provides a great support to represent context and deal with its dynamicity.

Figure 2 shows a simplified OWL implementation of the context model for the running business process example. From letf to right the figure shows some classes (such as *Product*, *ProductType*, *Customer* and *Seller*); some data properties (such as *name*, *price* and *inPromotion*)); some instances of the *ProductType* and *Product* classes; and the property values of the product instance *AirConditionrLex125_6*.

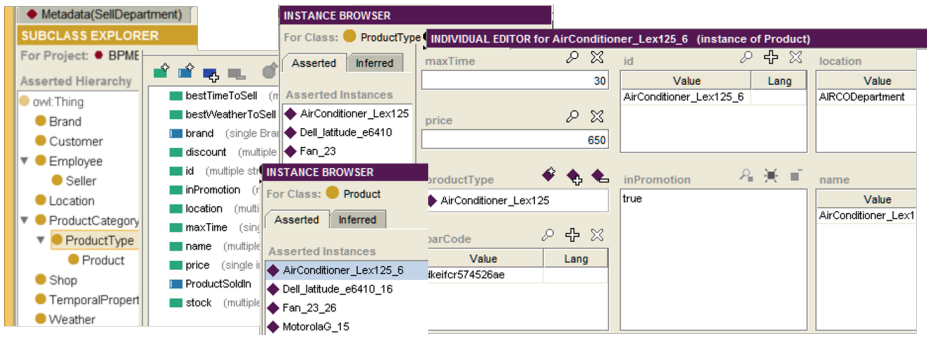

Fig. 2: Snapshot of the context model created for the running example.

3.2 CPN Extension

CPN is defined is as follows [1]:

- P the places.
- T the transitions.
- $F \subseteq (P \times T) \cup (T \times P)$ the arcs in the model.
- $\sum$ the color sets of the tokens.
- V the set of typed variables, such that $Type[v] \in \sum, \forall\, v \in V$.
- $C : P \rightarrow \sum$ the color set functions.
- $G : T \rightarrow EXPR_V$ a guard function, such that for each transition $t\, Type[G(t)] = Bool$.
- $E : F \rightarrow EXPR_V$ an arc expression function such that for each arc a $Type[E(a)] = C(p)_{MS}$ where p is the place connected to a.
- $I : P \rightarrow EXPR_\emptyset$ an initialization function that assigns an initializations expression to each p such that $Type[I(p)] = C(p)_{MS}$.

As such, CPN is a tuple $(\sum, P, T, F, C, V, G, E, I)$ with $EXPR$ the set of expressions provided by the inscription language, e.g., CPN ML. $Type[e]$ resembles the *type* of an expression $e \in EXPR$. Timing is considered as one of the color sets and is not explicitly introduced.

After analysing this definition, we determine that context can be introcuced in the following constructs in CPN:

- **Transition guards:** can contain context data to enforce dynamic conditions on tokens.
- **Transition timing:** context can express time constraints and conditions, which can be incorporated into the transitions.
- **Transition actions:** can contain functions whose parameters are context data
- **Arc expressions:** arcs determine the flow relationships between places and transitions and may have expressions that contain context data, for instance, for indicating a delay in the flow, for indicating the number of tokens, etc.
- **Place inicial marking:** can contain context data to enforce dynamic values on tokens.

By taking into account the extensions of these constructs to enable the use of context, our extension can be defined as follows:

- $\sum_A$ the context-aware color sets, which are defined in the context model.
- V_A the variables of the color sets for which $Type[v_a] \in \sum_A, \forall\, v_a \in V_A$. In this definition, all the variables are defined in the context model to obtain maximal flexibility.
- P_A the context-aware places for which $C_A : P_A \rightarrow \sum_A$ are the context-aware color set functions.
- T_A the context-aware transitions with $G_A : T_A \rightarrow EXPR_{V_A}$ their guards.
- F_A the context-aware arcs with $E_A : F_A \rightarrow EXPR_{V_A}$ their expressions.

By this definition, context is introduced as a special case of data-awareness in the CPN. It can cover all of the behavior which is represented by colors in the model. With $EXPR_{V_A}$, the context has its own way of incorporating data operations outside of the CP-net. This separates the workflow logic from the data logic, but connects them in a call-and-response approach. It is also possible to apply extra reasoning in the context model. For example, OWL provides great facilities for inferring new information by using, e.g., SWRL rules[2]. This extends the inscription language $EXPR_{V_A}$ to a more powerful subject in the workflow.

The context-awareness does not change the marking of transitions in a particular way, as we do not interfere with the occurrence rules. However, the extension is now able to retrieve the data from an instantiation of the context model, which allows the workflow to be aware of the current execution context and properly deal with the context dynamicity. Also, by using ontologies, it becomes possible to define data that can be used throughout different workflows of a system. Moreover, ontologies provide a very high expressiveness for defining data, which makes the data-awareness of the workflow much stronger; for instance, now it is possible to create complex classes that are interrelated, which is not supported in CPN.

[2] http://www.w3.org/Submission/SWRL/

4 Tool Support

In order to support the modeling and execution of context-adaptive business processes, we have extended CPN Tools [4]. Although there are other software packages such as, ExSpect [16] and GreatSPN [17], we chose CPN Tools because it has been widely used and supported, which results in timely updates and the introduction of new features such as the incorporation of declarative constraints and extensions. CPN Tools is a free of charge tool that provides state-space analysis, simulation, and the generation of event logs that can be used for further process analysis (e.g., process mining). The tool relies on the functional programming language *Meta Language* (ML) to incorporate simulation and analysis, and enables the user to implement custom functions and probability distributions that can be used throughout the model. The modeling tool that we have developed to manage context-adaptive business processes is composed by a Context Management Plug-in and an extension of CPN Tools.

4.1 Context Management Plugin

Context is managed as a separate concern in order to facilitate system design, reuse, and maintenance. We have developed a plug-in that allows the OWL model to be interpreted, updated, and saved. Specifically, the API allows opening and saving an OWL context model as well as managing its context instances (i.e., individuals) independently of their classes, such as methods for retrieving and updating data and object properties of a certain instance.

To implement this class, we have used Jena 2.4[3], and the Pellet reasoner 1.5.2[4]. Jena is a Java framework for building Semantic Web applications that provides a programmatic environment for creating, examining, and modifying an OWL ontology. Pellet is an open-source tool that provides reasoning services for OWL ontologies. Pellet facilitates accessing to the information stored in the ontology. For example, the *getDataPropertyBool()* method, which retrieves the value of a boolean data property, is implemented as follows:

```
public static boolean getDataPropertyBool(String
   individualID, String attributeID) {
        boolean attributeValue=false;
         try {
                 dataset.begin( ReadWrite.READ);
                 Individual ind=
                    ontModel.getIndividual(prefixURI
                    + individualID);
                 Property prop=
                    ontModel.getProperty(prefixURI +
                    attributeID);
                try{
```

[3] https://jena.apache.org/
[4] http://clarkparsia.com/pellet/

```
                attributeValue=
                   ind.getPropertyValue(prop).asLiteral().getBoolean();
                } catch(Exception e){
                        System.out.println("The
                           context property
                           identifiers are not
                           correct");
                }
          dataset.close();
    } finally {dataset.end();}

          return attributeValue;
}
```

4.2 CPN Tools Extension

We have developed a Java extension in CPN Tools 4.0 which recently added a framework for this purpose [4]. By implementing an extension, it is now possible to use Java methods in the modeled Petri nets by means of remote procedure calls (RPC). These calls reach the ML engine which executes them and inputs the returned values into the model. Thus, context represented in the OWL context model can be injected into the Petri net model as data by means of these RPCs. Using this functionality, we have implemented a CPN Tool extension, called CM, that provides methods to interpret and update the context model. These methods use the Context Management plug-in explained above, which contains the proper logic to manage the context model. This tool, ready for supporting the modeling and execution of context-adaptive business processes can be downloaded at https://perswww.kuleuven.be/∼u0095631/.

Using the presented extension, Figure 3 shows the context-aware enriched solution of the running example in Figure 3. Various ML functions are referring to the CM name space, which communicates with the Java code that calls OWL data. The specific ML functions that are shown in the figure are:

- *retrieveInstances(contextClassID)*: this function retrieves all instances of a particular context class. They are transformed by the ML function *deserialize* and inputted as tokens into the system as can be seen for customers, sellers and products.
- *contextProp{Int/Bool/String}(IndividualID,PropertyID)*: this function retrieves the value (integer, boolean or string) of the property for the entered context instance.
- *setContextProp{Int/Bool/String}(IndividualID,PropertyID,New value)*: this function sets the value (integer, boolean or string) of the property for the entered context instance.
- *ContextObjProp(IndividualID,PropertyID)*: this method retrieves the identifier of the object that is related with the *IndividualID* by the *PropertyID*.

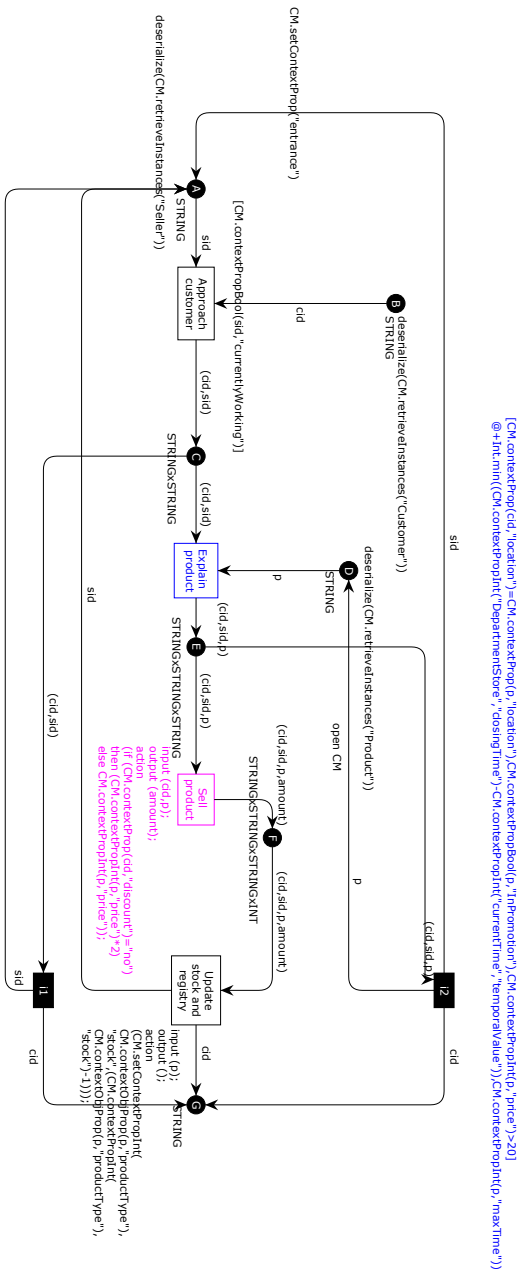

Fig. 3: Implementation example which extends the workflow of the shop with context data.

They are aligned with the Java methods, which implement the functionality of calling the OWL database. For example *contextPropBool()* is implemented as follows:

```
public boolean contextPropBool(String instanceID,
    String contextPropertyID){
         return
             contextModel.getDataPropertyBool(instanceID,
             contextPropertyID);
};
```

Thus, the running example is now expressed in a context-adaptive way. The id of the seller *sid* is now linked with the shifts of the individuals, which are retrieved in the guard of *Approach Customer*. The transition *Explain product* is bound heavily with various context constraints and are indicated in blue: the seller will offer guidance to customers when the product is in promotion, the price is higher than a certain amount (which can also be defined in the context), and the customer that is visiting the store is currently at the right location. Otherwise, this transition will be skipped and the customer leaves the system through the invisible transition. The seller and the product become available again. Also, the duration of the transition is dependent on the time that is left until closing time. The price of the product is calculated in *Sell product* and depends on the information that resides within the context of the customer. For instance, if the customer has already bought numerous products, s/he might benefit from a discount. Finally, the stock levels are changed in the net by updating the data of the corresponding product in the context model. This is an example of how the context gets changed by the model and offers two-way synchronization of the data. Overall, all the data contributions in the colored Petri net are made dynamic and semantically defined in the OWL ontology, which is superior to the static approach of CPN Tools for enactment. Previously, all the data had to be stored in the separate records within CPN Tools or a file where they could be retrieved from. In our approach, the data is inputted at runtime from the OWL model, where the context is stored in a semantic and highly expressive way.

5 Discussion

While CPN Tools offers great support for modeling and enacting CPN, not all the benefits of context adaptation can be leveraged by its implementation. The calculation of the current context state is done upon a change in the places surrounding a transition. As such, other transitions can manipulate the context and render it unfit for the enabling of the transition, while the context was already verified and enabled it. This also avoids inputting tokens into the CPN after its initialization.

In addition, the syntactics to incorporate context properties into the CPN are rather complex and perhaps not immediately user-friendly. This can be solved

by explicitly declaring the context in color sets. However, this creates the need of syncronizing the context model and the color set every time the net is executed.

Furthermore, the extensions framework of CPN tools is still constrained by the usage of the boolean, integer, and string (BIS) types, which limits the incorporation of high-level or generic data types. This implies that only simple context can be incorporated, and a separation between ontology variables and CPN variables exist. This boils down to a change in the definition to $A \subseteq \sum$ the color sets in the context with V_A the variables for the color sets consisting of $Type[v_a] \in Bool, String, Integer, \forall\, v_a \in V_a$. The two data typologies need to be merged and made compatible, which limits the implementation to manage the context by using the identifiers of each ontology element instead of the elements themselves; e.g., instead of getting an object of a product class and retrieving the information directly from the object, only its identifier is retrieved. Therefore, it is necessary that the context functions make a search to find the intance that matches the identifier in order to retrieve or modify the corresponding data.

Finally, CPN allow for state-space analysis, which enables modelers and users to search for unwanted behavior such as dead locks and infrequent or overly-frequent execution paths. This analysis would greatly benefit the study of enacting context in workflows. However, it is not straightforward to introduce state-space analysis in the presented tool, as the state of the context is not explicitly recognized by CPN tools. To perform state-space analysis, one has to retrieve a certain context upfront and incorporate it as data into the CPN. This somewhat circumvents the dynamics of context data, but can still achieve a rapid implementation for analyzing the effect of certain contexts in workflows.

6 Conclusions

In this paper, we have presented a tool that supports the management of context-adaptive business processes enabling their use in dynamic environments. The tool extends CPN Tools with ontologies in order to support the modeling and execution of context-adaptive CPN. Using the extended tool, the data that is already available in the information system context can be used in the Petri nets of the system with minimal effort; while using ontologies, the context of the system can be semantically represented and properly managed at runtime. Ontology-based context models also make it easy to navigate the data structure that is present in the information system and add it to the Petri nets.

As such, workflows can be enriched with the context defined in the context ontology models, and in the other way around, the process can also reach the context and update data that is encountered within the workflow.

Acknowledgments

This work has been funded by the KU Leuven Research Fund (F+ Fellowship J:BOF/MS/F+/13/032) and the FWO (Fonds voor Wetenschappelijk Onderzoek) Project G0804 13N.

References

1. Jensen, K.: Coloured petri nets. Springer (1987)
2. van der Aalst, W.M.: Making work flow: On the application of petri nets to business process management. In: Application and Theory of Petri Nets 2002. Springer (2002) 1–22
3. C. Rosemann, M., R.J.F.: Contextualisation of business processes. Int. J. Bus. Process Integr. Manag. **3**(1) (2008) 4760
4. Westergaard, M.: Cpn tools 4: multi-formalism and extensibility. In: Application and Theory of Petri Nets and Concurrency. Springer (2013) 400–409
5. Saidani, O., Nurcan, S.: Context-awareness for adequate business process modelling. In: RCIS. (2009) 177186
6. Vara, J.D.L., Ali, R., Dalpiaz, F.: Business processes contextualisation via context analysis. In: ER. (2010) 471476
7. Goedertier, S., J. Vanthienen, title = Compliant and flexible business processes with business rules, b..B.a.C.y...p...
8. Dadam, P., Reichert, M.: The adept project: a decade of research and development for robust and exible process support. challenges and achievements. Comput. Sci. Dev. (2009) 8196
9. Lu, T., Bao, J.: A systematic approach to context aware service design. Journal of Computers **7**(1) (2012)
10. Tang, F., Guo, M., Dong, M., Li, M., Guan, H.: Towards context-aware workflow management for ubiquitous computing. In: Embedded Software and Systems, 2008. ICESS'08. International Conference on, IEEE (2008) 221–228
11. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: A framework for the management of context-aware workflow systems. In: WEBIST (1). (2007) 80–87
12. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: Context-aware workflow management. In: Web Engineering. Springer (2007) 47–52
13. Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review (2004) 197–207
14. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing (2007)
15. Ye, J., Coyle, L., Dobson, S., Nixon, P.: Ontology-based models in pervasive computing systems. The Knowledge Engineering Review **22:4** (2007) 315–347
16. van der Aalst, W.M., de Crom, P.J., Goverde, R.R., van Hee, K.M., Hofman, W.J., Reijers, H.A., van der Toorn, R.A.: Exspect 6.4 an executable specification tool for hierarchical colored petri nets. In: Application and Theory of Petri Nets 2000. Springer (2000) 455–464
17. Baarir, S., Beccuti, M., Cerotti, D., De Pierro, M., Donatelli, S., Franceschinis, G.: The greatspn tool: recent enhancements. ACM SIGMETRICS Performance Evaluation Review **36**(4) (2009) 4–9